

Guaranteed Bounds for General Approximate Dynamic Programming

Yajing Liu, Edwin K. P. Chong, Ali Pezeshki, and Bill Moran

Abstract—In this paper, we will develop a systematic approach to deriving guaranteed bounds for approximate dynamic programming (ADP) schemes in optimal control problems. Our approach is inspired by our recent results on bounding the performance of greedy strategies in optimization of string-submodular functions over a finite horizon. The approach is to derive a string-submodular optimization problem, for which the optimal strategy is the optimal control solution and the greedy strategy is the ADP solution. Using this approach, we show that any ADP solution achieves a performance that is at least a factor of β of the performance of the optimal control solution, which satisfies Bellman’s optimality principle. The factor β depends on the specific ADP scheme, as we will explicitly characterize. To illustrate the applicability of our bounding technique, we present examples of ADP schemes, including the popular rollout method.

I. INTRODUCTION

In sequential decision making, adaptive sensing, and adaptive control, we are frequently faced with optimally choosing a string (finite sequence) of actions over a finite horizon to maximize an objective function. However, computing the optimal strategy (optimal sequence of actions) is often difficult. One approach is to use dynamic programming via Bellman’s principle for optimality (see, e.g., [3]). However, the computational complexity of this approach grows exponentially with the size of the action space and the decision horizon. Because of this inherent complexity, for years, there has been interest in developing approximation methods for solving dynamic programming problems. Although a wide range of approximate dynamic programming (ADP) methods have been developed (see, e.g., [14]), a general systematic technique to provide performance guarantees for them has remained elusive. In this paper, we will develop a systematic approach to deriving guaranteed bounds for ADP schemes. Our approach is inspired by our recent results in [22] and [23]) on bounding the performance of greedy strategies in optimization of string-submodular functions.

Submodularity of functions over finite sets plays an important role in discrete optimization (see, e.g., [12], [13], [5], [18], [16], [2], [19], [20], [6], [7], [15], [21], [1], [9],

and [10]). It has been shown that, under submodularity, the greedy strategy provides at least a constant-factor approximation to the optimal strategy. For example, the celebrated result of Nemhauser *et al.* [13] states that for maximizing a monotone submodular function F over a uniform matroid such that $F(\emptyset) = 0$ (here \emptyset denotes the empty set), the value of the greedy strategy is no less than a factor $(1 - e^{-1})$ of that of the optimal strategy. This is a powerful result. But a drawback is that submodular functions studied in most previous papers are defined on the power set of a given finite set. In contrast, in adaptive control and sensing, we are interested in choosing a string of action sequentially, and the value of the objective function depends on the *order* of these actions. In consequence, we cannot apply the result of Nemhauser *et al.* [13] or its related results on submodularity over finite sets.

To compare the greedy and optimal strategies for functions defined over strings, in [22] and [23], we have introduced the notion of *string*-submodularity, which builds on the notion of set-submodularity in combinatorial optimization. We have shown that, under string-submodularity, any greedy strategy is suboptimal by a factor of at worst $(1 - e^{-1})$, entirely consistent with the result of Nemhauser *et al.* [13]. Our framework also includes characterizing the *curvature* of string-submodular functions, which roughly corresponds to the quantitative “degree” of submodularity. In fact, there are several notions of curvature (to be described later). Subject to curvature, we have derived suboptimality bounds for greedy strategies that are strictly better than $(1 - e^{-1})$. These results represent the state-of-the-art in bounding greedy strategies in string-submodular optimization problems.

In this paper, inspired by the bounding techniques in [22] and [23], we develop the first systematic approach to deriving performance bounds for general ADP methods for optimal control problems. To set up our approach, in Section II, we review our string-submodularity results, notions of curvature, and the corresponding bounds from. In Section III, we first describe a general optimal control problem and a class of ADP schemes for approximating optimal control solutions. We then describe our approach to bounding the performance of such ADP schemes. The idea is to define a string-submodular optimization problem for which the optimal strategy is the optimal control solution, and the greedy strategy is the ADP solution. Though, inspired by our previous work, the bounding of ADP schemes is based on a new technique for general string-optimization problems. The results in Section II, simply set the stage and terminology for new developments and results that we will present in Section IV. We show that any ADP solution achieves a performance

This work is supported in part by NSF under Grant CCF-1018472, and by AFOSR under Grant FA9550-12-1-0418, and by CSU Information Science and Technology Center (ISTeC).

Y. Liu is with the Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523, USA yajing.liu@ymail.com

E. K. P. Chong and A. Pezeshki are with the Department of Electrical and Computer Engineering, and the Department of Mathematics, Colorado State University, Fort Collins, CO 80523, USA Edwin.Chong, Ali.Pezeshki@Colostate.Edu

B. Moran is with the Department of Electrical and Electronic Engineering, The University of Melbourne, Melbourne, VIC 3010, Australia wmoran@unimelb.edu.au

that is at least a factor of β of the performance of the optimal control solution (satisfying Bellman's optimality principle). The factor β depends on the specific ADP scheme, in way that we will explicitly characterize. In Section V, we present a few examples of ADP schemes to illustrate the application of our results. In particular, we consider rollout policies which represent a well-studies family of ADP schemes (see, e.g., [4]). Finally, in Section VI, we present our concluding remarks.

II. STRING-SUBMODULARITY AND PERFORMANCE BOUNDS FOR GREEDY STRATEGIES

In this section, we review our string-submodularity results, notions of curvature, and the corresponding bounds from [22] and [23]. These results show that greedy strategies for optimizing a string-submodular function achieve at least a factor of α of the performance of optimal strategies, which are characterized by Bellman's optimality principle. The factor α depends on the specific objective function to be optimized and its various curvatures, but it is at least $(1 - e^{-1})$. The results presented here set the stage, terminology, and the inspiration for our new developments in Section IV for bounding the performance of ADP schemes.

A. String-Submodularity and Curvatures

Let \mathbb{A} be a set of possible actions. At each stage i , we choose an action a_i from \mathbb{A} . Let $A = (a_1, a_2, \dots, a_k)$ be a *string* of actions taken over k consecutive stages, where $a_i \in \mathbb{A}$ for $i = 1, 2, \dots, k$. Let $\mathbb{A}^* = \{(a_1, a_2, \dots, a_k) \mid k = 0, 1, \dots \text{ and } a_i \in \mathbb{A}, i = 1, 2, \dots, k\}$ be the set of all possible strings of actions. Note that $k = 0$ corresponds to the empty string (no action taken), denoted by \emptyset .

For a given string $A = (a_1, a_2, \dots, a_k)$, we define its *string length* as k , denoted $|A| = k$. If $M = (a_1^m, a_2^m, \dots, a_{k_1}^m)$ and $N = (a_1^n, a_2^n, \dots, a_{k_2}^n)$ are two strings in \mathbb{A}^* , we say $M = N$ if $|M| = |N|$ and $a_i^m = a_i^n$ for each $i = 1, 2, \dots, |M|$. Moreover, we define string *concatenation* as $M \oplus N = (a_1^m, a_2^m, \dots, a_{k_1}^m, a_1^n, a_2^n, \dots, a_{k_2}^n)$. If M and N are two strings in \mathbb{A}^* , we write $M \preceq N$ if we have $N = M \oplus L$, for some $L \in \mathbb{A}^*$. In other words, M is a *prefix* of N .

String Submodularity. A function from strings to real numbers, $f : \mathbb{A}^* \rightarrow \mathbb{R}$, is *string submodular* if

- i. f has the *forward-monotone* property, i.e., $\forall M \preceq N \in \mathbb{A}^*, f(M) \leq f(N)$.
- ii. f has the *diminishing-return* property, i.e., $\forall M \preceq N \in \mathbb{A}^*, \forall a \in \mathbb{A}, f(M \oplus (a)) - f(M) \geq f(N \oplus (a)) - f(N)$.

We assume, without loss of generality, that $f(\emptyset) = 0$. Otherwise, we can replace f with the marginalized function $f - f(\emptyset)$. From the forward-monotone property, we know that $f(M) \geq 0$ for all $M \in \mathbb{A}^*$.

Curvatures. We define several notions of curvature for f as follows.

- 1) *Total backward curvature of f :*

$$\sigma = \max_{a \in \mathbb{A}, M \in \mathbb{A}^*} \left\{ 1 - \frac{f((a) \oplus M) - f(M)}{f((a)) - f(\emptyset)} \right\}.$$

- 2) *Total backward curvature of f with respect to string $M \in \mathbb{A}^*$:*

$$\sigma(M) = \max_{N \in \mathbb{A}^*, 0 < |N| \leq K} \left\{ 1 - \frac{f(N \oplus M) - f(M)}{f(N) - f(\emptyset)} \right\}.$$

- 3) *Total forward curvature of f :*

$$\epsilon = \max_{a \in \mathbb{A}, M \in \mathbb{A}^*} \left\{ 1 - \frac{f(M \oplus (a)) - f(M)}{f((a)) - f(\emptyset)} \right\}.$$

- 4) *Total forward curvature of f with respect to M :*

$$\epsilon(M) = \max_{N \in \mathbb{A}^*, 0 < |N| \leq K} \left\{ 1 - \frac{f(M \oplus N) - f(M)}{f(N) - f(\emptyset)} \right\}.$$

- 5) *Elemental forward curvature of f :*

$$\eta = \max_{a_i, a_j \in \mathbb{A}, M \in \mathbb{A}^*} \frac{f(M \oplus (a_i) \oplus (a_j)) - f(M \oplus (a_i))}{f(M \oplus (a_j)) - f(M)}.$$

B. Performance Bounds for Greedy Strategies

Consider the problem of finding a string $M \in \mathbb{A}^*$, with a length $|M|$ not larger than K (prespecified), to maximize the objective function f , that is

$$\begin{aligned} & \text{maximize } f(M) \\ & \text{subject to } M \in \mathbb{A}^*, |M| \leq K. \end{aligned} \quad (1)$$

We define optimal and greedy strategies for (1) as follows:

- (1) *Optimal strategy:* Consider the problem (1) of finding a string that maximizes f under the constraint that the string length is not larger than K . We call a solution of this problem an *optimal strategy* (a term we already have used repeatedly before). Note that if the function f is forward monotone and there exists an optimal strategy, then there exists one with length K .
- (2) *Greedy strategy:* A string $G_k = (g_1, g_2, \dots, g_k)$ is called *greedy* if $\forall i = 1, 2, \dots, k$,

$$g_i \in \operatorname{argmax}_{g \in \mathbb{A}} f((g_1, g_2, \dots, g_{i-1}, g)),$$

where argmax denotes the set of actions that maximize $f((g_1, g_2, \dots, g_{i-1}, g))$.

Let I be the subset of \mathbb{A}^* with maximal string length K : $I = \{A \in \mathbb{A}^* : |A| \leq K\}$. We call I a *uniform structure*. Note that the way we define uniform structures is similar to the way independent sets associated with uniform matroids are defined. We now present the relationship between total curvatures and approximation bounds for the greedy strategy.

Theorem 1: [22] (Greedy approximation bounds involving total curvatures). Consider a string submodular function f . Let O be a solution to (1). Then, any greedy string G_K satisfies

- (i)

$$\begin{aligned} f(G_K) & \geq \frac{1}{\sigma(O)} \left(1 - \left(1 - \frac{\sigma(O)}{K} \right)^K \right) f(O) \\ & > \frac{1}{\sigma(O)} (1 - e^{-\sigma(O)}) f(O), \end{aligned}$$

- (ii)

$$f(G_K) \geq (1 - \max_{i=1, \dots, K-1} \epsilon(G_i)) f(O).$$

Under the framework of maximizing submodular set functions, similar results are reported in [5]. However, the forward and backward algebraic structures are not exposed in [5] because the total curvature there does not depend on the order of the elements in a set. In the setting of maximizing string submodular functions, the above theorem exposes the roles of forward and backward algebraic structures in bounding the greedy strategy.

The results in Theorem 1 imply that for a string submodular function, we have $\sigma(O) \geq 0$. Otherwise, part (i) of Theorem 1 would imply that $f(G_K) \geq f(O)$, which is absurd. Moreover, recall that if the function is backward monotone, then $\sigma(O) \leq \sigma \leq 1$ and we have the following result.

Corollary 1: [22] (Universal greedy approximation bounds involving total curvatures). Suppose that f is string-submodular and backward monotone. Then,

(i)

$$\begin{aligned} f(G_K) &\geq \frac{1}{\sigma} \left(1 - \left(1 - \frac{\sigma}{K} \right)^K \right) f(O) \\ &> \frac{1}{\sigma} (1 - e^{-\sigma}) f(O), \end{aligned}$$

(ii)

$$f(G_K) \geq (1 - \epsilon) f(O).$$

Note that the bounds $\frac{1}{\sigma}(1 - e^{-\sigma})$ and $(1 - \epsilon)$ are independent of the length constraint K . Therefore, the above bounds are universal lower bounds for the greedy strategy for all possible length constraints. Part (i) of Corollary 1 implies that in the backward monotone case, where $\sigma \leq 1$, any greedy string G_K satisfies the universal bound $f(G_K) > (1 - e^{-1})f(O)$.

Theorem 2: [22] (Greedy approximation bounds involving elemental curvature). Consider a forward-monotone function f with elemental forward curvature η . Let O be an optimal solution to (1). Suppose that $f(G_i \oplus O) \geq f(O)$ for $i = 1, 2, \dots, K - 1$. Then, any greedy string G_K satisfies

$$f(G_K) \geq f(O) \left(1 - \left(1 - \frac{1}{K_\eta} \right)^K \right),$$

where $K_\eta = (1 - \eta^K)/(1 - \eta)$ if $\eta \neq 1$ and $K_\eta = K$ if $\eta = 1$.

Recall that η does not depend on the length constraint K . Therefore, the lower bound using K_η is a universal lower bound for the greedy strategy. Now suppose that f is string submodular. Then, we have $\eta \leq 1$. Because $1 - (1 - \frac{1}{K_\eta})^K$ is decreasing as a function of η , an immediate consequence of Theorem 2 is that any greedy string G_K satisfies the universal bound $f(G_K) > (1 - e^{-1})f(O)$.

C. Other Results

In the previous section, we considered the case where I is a uniform structure. In [22] and [23], we have also studied the case where I is a *non-uniform* structure, by introducing the notion of string-matroid, and have derived bounds that quantify the performance of greedy strategies

relative to optimal strategies in terms of various curvatures of the objective function. We leave these results out for the sake of brevity and refer the reader to [22] and [23] for details.

A number of other researchers (see [16], [2], and [8]) have also considered bounding the performance of greedy strategies using extensions of set submodularity to string-submodularity. In particular, Streeter and Golovin [16] showed that if the function f is *forward* and *backward* monotone: $f(M \oplus N) \geq f(M)$ and $f(M \oplus N) \geq f(N)$ for all $M, N \in \mathbb{A}^*$, and f has the diminishing-return property: $f(M \oplus (a)) - f(M) \geq f(N \oplus (a)) - f(N)$ for all $a \in \mathbb{A}$, $M, N \in \mathbb{A}^*$ such that M is a prefix of N , then the greedy strategy achieves at least a $(1 - e^{-1})$ -approximation of the optimal strategy. However, the notions of string submodularity and various curvature that we have introduced in our recent work [22], [23] provide us with weaker sufficient conditions under which the greedy strategy still achieves at least a $(1 - e^{-1})$ -approximation of the optimal strategy.

III. BOUNDING ADP SCHEMES IN OPTIMAL CONTROL

In this section, we first describe a general optimal control problem and a class of ADP schemes for approximating optimal control solutions. We then describe our approach to bounding the performance of such ADP schemes.

A. General Optimal Control Problems

To begin our formulation of a general optimal control problem, let \mathcal{X} denote a set of states and \mathcal{A} a set of control actions. Given $x_1 \in \mathcal{X}$ and functions $h_k : \mathcal{X} \times \mathcal{A} \rightarrow \mathcal{X}$ and $r_k : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}_+$ for $k = 1, \dots, K$, consider the optimal control problem

$$\begin{aligned} &\text{maximize}_{a_1, \dots, a_K \in \mathcal{A}} \quad \sum_{k=1}^K r_k(x_k, a_k) \\ &\text{subject to} \quad x_{k+1} = h_k(x_k, a_k), \quad k = 1, \dots, K-1. \end{aligned} \quad (2)$$

Think of a_k as the *control action* applied at time k and x_k the *state* visited at time k . The real number $r_k(x_k, a_k)$ is the *reward* accrued at time k by applying a_k at state x_k . This form of optimal control problem covers a wide variety of optimization problems found in many areas, ranging from engineering to economics. In particular, many adaptive sensing problems have this form (see, e.g., [11]).

The solution to the optimal control problem above is characterized by Bellman's principle of dynamic programming. To explain, for each $k = 1, \dots, K$, define functions $V_k : \mathcal{X} \times \mathcal{A}^{K-k+1} \rightarrow \mathbb{R}_+$ by

$$V_k(x_k, (a_k, \dots, a_K)) = \sum_{i=k}^K r_i(x_i, a_i)$$

where $x_{i+1} = h_i(x_i, a_i)$, $i = k, \dots, K-1$. The optimal control problem can be written as

$$\begin{aligned} &\text{maximize}_{a_1, \dots, a_K \in \mathcal{A}} \quad V_1(x_1, (a_1, \dots, a_K)) \\ &\text{subject to} \quad x_{k+1} = h_k(x_k, a_k), \quad k = 1, \dots, K-1, \end{aligned}$$

where x_1 is given. Let o_1, \dots, o_K be an optimal solution to this problem, and given x_1 , define $x_1^* = x_1$ and $x_{k+1}^* = h_k(x_k^*, o_k)$, $k = 1, \dots, K-1$. This is the sequence of states visited as a result of the optimal control actions o_1, \dots, o_K . Then, Bellman's principle states that for $k = 1, \dots, K$, we have

$$\begin{aligned} V_k(x_k^*, (o_k, \dots, o_K)) = \\ \max_{a \in \mathcal{A}} \{ r_k(x_k^*, a) + V_{k+1}(h_k(x_k^*, a), (o_{k+1}, \dots, o_K)) \}, \\ o_k \in \\ \operatorname{argmax}_{a \in \mathcal{A}} \{ r_k(x_k^*, a) + V_{k+1}(h_k(x_k^*, a), (o_{k+1}, \dots, o_K)) \}, \end{aligned} \quad (3)$$

with the convention that $V_{K+1}(\cdot) \equiv 0$. Moreover, any sequence of control actions satisfying (3) above is optimal. The term $V_{k+1}(h_k(x_k^*, a), (o_{k+1}, \dots, o_K))$ is called the *value-to-go* (VTG).

Bellman's principle provides a method to compute an optimal solution: We use (3) to iterate backwards over the time indices $k = K, K-1, \dots, 1$, keeping the states as variables, working all the way back to $k = 1$. This is the familiar *dynamic programming algorithm*. However, the procedure suffers from the *curse of dimensionality* and is therefore impractical for many problems of interest: merely storing the iterates $V_k(\cdot, (o_k, \dots, o_K))$ requires an exponential amount of memory. Therefore, designing computationally tractable approximation methods remains a topic of active research.

B. ADP Schemes

A well-studied class of *approximate dynamic programming* (ADP) approaches rests on approximating the VTG $V_{k+1}(h_k(x_k^*, a), (o_{k+1}, \dots, o_K))$ by some other term $W_{k+1}(\hat{x}_k, a)$. In this method, we start at time $k = 1$, at state $\hat{x}_1 = x_1$, and for each $k = 1, \dots, K$, we compute the control action and state using

$$\begin{aligned} \hat{a}_k &\in \operatorname{argmax}_{a \in \mathcal{A}} \{ r_k(\hat{x}_k, a) + W_{k+1}(\hat{x}_k, a) \}, \\ \hat{x}_{k+1} &= h_k(\hat{x}_k, \hat{a}_k). \end{aligned} \quad (4)$$

The VTG approximation $W_{k+1}(\hat{x}_k, a)$ can be based on a number of methods, ranging from heuristics to reinforcement learning [17] to rollout [4].

A natural question is “what is the performance of the ADP approach above relative to the optimal solution?” The answer, of course, depends on the specific VTG approximation. If the VTG approximation is equal to the true VTG, then the procedure above generates an optimal solution. In general, the procedure produces something suboptimal. But how suboptimal? This question has alluded general treatment but has remained an issue of great interest to designers and users of ADP methods. In the following section, we develop a systematic approach to answering this fundamental question.

C. Deriving Performance Bounds for ADP Schemes

We now describe our approach to bounding the performance of such ADP schemes. The idea is to define a string-submodular optimization problem for which the optimal strategy is the optimal control solution, and the greedy strategy is the ADP solution. Though inspired by

our previous work (reviewed in Section II), the bounding of ADP schemes is based on a new technique for general string-optimization problems.

To see how our approach works, let \mathcal{A}_k be the set of all strings of symbols in \mathcal{A} with length not exceeding k . Define the function $f : \mathcal{A}_K \rightarrow \mathbb{R}_+$ by $f(\varnothing) = 0$ and

$$f((a_1, a_2, \dots, a_k)) = \sum_{i=1}^k r_i(x_i, a_i) + W_{k+1}(x_k, a_k),$$

for $k = 1, \dots, K$, where $x_{k+1} = h_k(x_k, a_k)$ as before and $W_{K+1}(\cdot) \equiv 0$ by convention. Using this string function f , we can now define the optimization problem of finding a string (a_1, \dots, a_K) to maximize $f((a_1, \dots, a_K))$. This is an instance of the string-optimization problem described earlier.

It is clear that $f((a_1, \dots, a_K)) = \sum_{i=1}^K r_i(x_i, a_i)$, which is the objective function in (2). Hence, the string-optimization problem defined above is equivalent to the optimal control problem (2). Next, notice that a greedy scheme by definition has the following form, given (g_1, \dots, g_{k-1}) :

$$\begin{aligned} g_k &\in \operatorname{argmax}_{g \in \mathcal{A}} f((g_1, \dots, g_{k-1}, g)) \\ &\in \operatorname{argmax}_{g \in \mathcal{A}} \left\{ \sum_{i=1}^{k-1} r_i(x_i, g_i) + r_k(x_k, g) + W_{k+1}(x_k, g) \right\} \\ &\in \operatorname{argmax}_{g \in \mathcal{A}} \{ r_k(x_k, g) + W_{k+1}(x_k, g) \}. \end{aligned} \quad (5)$$

This is simply the ADP scheme in (4). Hence, we have the following result.

Proposition 1: The ADP scheme in (4) is a greedy strategy for the string-optimization problem

$$\begin{aligned} &\text{maximize } f((a_1, a_2, \dots, a_K)) \\ &\text{subject to } (a_1, a_2, \dots, a_K) \in \mathcal{A}_K. \end{aligned} \quad (6)$$

Using this proposition, we can show that any ADP solution achieves a performance that is at least a factor of β of the performance of the optimal control solution (satisfying Bellman's optimality principle). The factor β depends on the specific ADP scheme as we will explicitly show in the next section.

IV. MAIN RESULTS

A. General Bound

In the last section, we introduced a general optimal control problem and an associated class of ADP schemes. We then formulated a string-optimization problem associated with a given optimal control problem and ADP scheme with the property that any optimal strategy for the string-optimization problem is an optimal control solution and any greedy strategy is the ADP solution. This allows us to use bounding methods for greedy strategies for string-optimization to derive bounds for ADP methods. However, it turns out that the results in Section II do not directly apply to the string-optimization problem we formulated in Section III. More specifically, the function f in Section III is defined only on \mathcal{A}_K (i.e., strings of length at most K), whereas the results in Section II require f to be defined on

strings of length greater than K . To address this issue, we now present a *new* result for bounding greedy strategies for string-optimization problems.

Let $f : \mathcal{A}_K \rightarrow \mathbb{R}_+$ be an objective function. Consider the optimization problem

$$\text{maximize } f(S) \text{ subject to } S \in \mathcal{A}_K, |S| = K. \quad (7)$$

Let $O_K = (o_1, \dots, o_K)$ be optimal for (7). Let $G_K = (g_1, \dots, g_K)$ be a greedy strategy for (7), defined as before: given g_1, \dots, g_{k-1} ,

$$g_k \in \operatorname{argmax}_{g \in \mathbb{A}} f((g_1, \dots, g_{k-1}, g)). \quad (8)$$

As before, write $G_0 = O_0 = \emptyset$ and for $k = 1, \dots, K$, $G_k = (g_1, \dots, g_k)$ and $O_k = (o_1, \dots, o_k)$.

Inspired by the results in Section II, define the *forward curvature of f with respect to G_k* by

$$\epsilon_k = 1 - \frac{f(G_{k+1}) - f(G_k)}{f((g_1)) - f(\emptyset)}, \quad 0 \leq k \leq K-1. \quad (9)$$

Notice that $\epsilon_0 = 0$. Next, define the *elemental forward curvature of f with respect to O_k* by

$$\eta_k = \frac{f(O_{k+1}) - f(O_k)}{f(o_{k+1}) - f(\emptyset)}, \quad 0 \leq k \leq K-1. \quad (10)$$

Notice that $\eta_0 = 1$. We now present a result that bounds $f(G_K)$ relative to $f(O_K)$, using the definitions above.

Theorem 3: The following bound holds: $f(G_K) \geq \beta f(O_K)$, where

$$\beta = \frac{\sum_{i=0}^{K-1} (1 - \epsilon_i)}{\sum_{i=0}^{K-1} \eta_i}.$$

Proof: Using the definition of the forward curvature of f with respect to G_k , we have

$$\begin{aligned} f(G_2) - f(G_1) &= (1 - \epsilon_1)f(G_1), \\ f(G_3) - f(G_2) &= (1 - \epsilon_2)f(G_1), \\ &\vdots \\ f(G_k) - f(G_{k-1}) &= (1 - \epsilon_{k-1})f(G_1), \\ &\vdots \\ f(G_K) - f(G_{K-1}) &= (1 - \epsilon_{K-1})f(G_1), \end{aligned}$$

which leads to

$$f(G_K) = \sum_{i=0}^{K-1} (1 - \epsilon_i)f(G_1). \quad (11)$$

By the definition of elemental forward curvature of f with respect to O_k , we have

$$\begin{aligned} f(O_K) &= \sum_{i=1}^K (f((o_1, \dots, o_i)) - f((o_1, \dots, o_{i-1}))) \\ &= \eta_0 f(o_1) + \eta_1 f(o_2) + \dots + \eta_{K-1} f(o_K) \\ &\leq \sum_{i=0}^{K-1} \eta_i f(G_1). \end{aligned}$$

where $f(G_1) \geq f(a)$ for any $a \in \mathcal{A}$ by (8). Therefore,

$$f(G_1) \geq \frac{1}{\sum_{i=0}^{K-1} \eta_i} f(O_K). \quad (12)$$

Combining (11) and (12), we get

$$f(G_K) \geq \frac{\sum_{i=0}^{K-1} (1 - \epsilon_i)}{\sum_{i=0}^{K-1} \eta_i} f(O_K)$$

as desired. ■

Remarks:

1. Notice that the bound above holds without any assumption on the monotonicity of f . However, the bound is only meaningful if $\beta \geq 0$. A sufficient condition for this is the monotonicity of f . More precisely, if f is forward monotone with respect to G_k , then $\epsilon_k \leq 1$ for each k , and $\epsilon_0 = 0$, in which case $\beta > 0$.
2. It is easy to check that if

$$\sum_{i=0}^{K-1} \epsilon_i + \eta_i \leq K,$$

then $f(G_K) = f(O_K)$; i.e., the greedy strategy is optimal.

B. Bounding ADP Schemes

We can now apply the result of Theorem 3 to the function f defined in Section III. Doing so will provide bounds on general ADP schemes relative to optimal control solutions. To begin, recall that

$$f((a_1, \dots, a_k)) = \sum_{i=1}^k r_i(x_i, a_i) + W_{k+1}(x_k, a_k).$$

Assume without loss of generality that f is a nonnegative function (for otherwise, we can simply add a constant to each W_{k+1} term). For this form of f , we have

$$\epsilon_k = 1 - \frac{r_{k+1}(x_{k+1}, g_{k+1}) + W_{k+2}(x_{k+1}, g_{k+1}) - W_{k+1}(x_k, g_k)}{r_1(x_1, g_1) + W_2(x_1, g_1)},$$

and

$$\eta_k = \frac{r_{k+1}(x_{k+1}, o_{k+1}) + W_{k+2}(x_{k+1}, o_{k+1}) - W_{k+1}(x_k, o_k)}{r_1(x_1, o_{k+1}) + W_2(x_1, o_{k+1})}$$

Hence, applying Theorem 3, we have that for the ADP scheme G_K , $f(G_K) \geq \beta f(O_K)$ where β is related to the above ϵ_k and η_k as given in Theorem 3. In the next section, we provide some examples of special cases to illustrate this bound.

V. EXAMPLES

A. Rollout

For the remainder of the paper, assume that x_1 is a given state. Suppose that $\pi_b : \mathcal{X} \rightarrow \mathcal{A}$ is a given policy. Consider the associated ADP where $W_{k+1}(x_k, g) = \sum_{i=k+1}^K r_i(x_i, \pi_b(x_i))$, where $x_{k+1} = h_k(x_k, g)$ and $x_{i+1} = h_k(x_i, \pi_b(x_i))$ for $k+1 \leq i \leq K-1$. This ADP method is called *rollout* [4]; the policy π_b is called the *base policy*. For rollout, we have

$$\epsilon_k = 1 - \frac{r_{k+1}(x_{k+1}, g_{k+1}) + R_1 - R_2}{r_1(x_1, g_1) + \sum_{i=2}^K r_i(\tilde{x}_i, \pi_b(\tilde{x}_i))}, \quad (13)$$

where

$$\begin{aligned} R_1 &= \sum_{i=k+2}^K r_i(x_i, \pi_b(x_i)), \\ R_2 &= \sum_{i=k+1}^K r_i(\hat{x}_i, \pi_b(\hat{x}_i)), \\ x_{i+1} &= h_i(x_i, g_i) \text{ for } 1 \leq i \leq k+1, \\ x_{i+1} &= h_i(x_i, \pi_b(x_i)) \text{ for } k+2 \leq i \leq K-1, \\ \hat{x}_{k+1} &= h_k(x_k, g_k), \\ \hat{x}_{i+1} &= h_i(\hat{x}_i, \pi_b(\hat{x}_i)) \text{ for } k+1 \leq i \leq K-1, \\ \tilde{x}_2 &= h_1(x_1, g_1), \\ \tilde{x}_{i+1} &= h_i(\tilde{x}_i, \pi_b(\tilde{x}_i)) \text{ for } 2 \leq i \leq K-1. \end{aligned}$$

Moreover, we have

$$\eta_k = \frac{r_{k+1}(x_{k+1}, o_{k+1}) + R_3 - R_4}{r_1(x_1, o_{k+1}) + \sum_{i=2}^K r_i(\tilde{x}_i, \pi_b(\tilde{x}_i))}, \quad (14)$$

where

$$\begin{aligned} R_3 &= \sum_{i=k+2}^K r_i(x_i, \pi_b(x_i)), \\ R_4 &= \sum_{i=k+1}^K r_i(\hat{x}_i, \pi_b(\hat{x}_i)), \\ x_{i+1} &= h_i(x_i, o_i) \text{ for } 1 \leq i \leq k+1, \\ x_{i+1} &= h_i(x_i, \pi_b(x_i)) \text{ for } k+2 \leq i \leq K-1, \\ \hat{x}_{k+1} &= h_k(x_k, o_k), \\ \hat{x}_{i+1} &= h_i(\hat{x}_i, \pi_b(\hat{x}_i)) \text{ for } k+1 \leq i \leq K-1, \\ \tilde{x}_2 &= h_1(x_1, o_{k+1}), \\ \tilde{x}_{i+1} &= h_i(\tilde{x}_i, \pi_b(\tilde{x}_i)) \text{ for } 2 \leq i \leq K-1. \end{aligned}$$

We now show that for rollout, the function f is forward monotone with respect to G_k , which implies that $\epsilon_k \leq 1$ and $\beta > 0$ (see Remark 1 in Section IV).

Theorem 4: In rollout, $f(G_k) \geq f(G_{k-1})$ for $k = 1, \dots, K$.

Proof: We have

$$\begin{aligned} f(G_k) - f(G_{k-1}) &= f((g_1, \dots, g_{k-1}, g_k)) - f((g_1, \dots, g_{k-1})) \\ &= \left(\sum_{i=1}^{k-1} r_i(x_i, g_i) + r_k(x_k, g_k) + W_{k+1}(x_k, g_k) \right) - \\ &\quad \left(\sum_{i=1}^{k-1} r_i(x_i, g_i) + r_k(x_k, \pi_b(x_k)) + W_{k+1}(x_k, \pi_b(x_k)) \right) \\ &= (r_k(x_k, g_k) + W_{k+1}(x_k, g_k)) - \\ &\quad (r_k(x_k, \pi_b(x_k)) + W_{k+1}(x_k, \pi_b(x_k))). \end{aligned}$$

By (5), we have that

$$r_k(x_k, g_k) + W_{k+1}(x_k, g_k) = \max_{g \in \mathcal{A}} \{r_k(x_k, g) + W_{k+1}(x_k, g)\},$$

which implies that $f(G_k) \geq f(G_{k-1})$. \blacksquare

B. Rollout with Optimal Base Policy

Suppose that the base policy is the optimal policy. In this case, the VTG approximation term W_{k+1} is equal to the true VTG. As pointed out in Section III, the resulting rollout scheme is optimal and satisfies Bellman's optimality principle. In this case, of course $f(G_K) = f(O_K)$. To illustrate that the bound in Theorem 3 is tight in this case, we will show that $\beta = 1$. We do this by showing that $\sum_{i=0}^{K-1} \epsilon_i + \eta_i \leq K$ (see Remark 2 in Section IV). To see this, by (9), we have $\epsilon_0 = 0$ and $\epsilon_k = 1$ for $1 \leq k \leq K-1$. By (10), we have $\eta_0 = 1$ and $\eta_k = 0$ for $1 \leq k \leq K-1$. Therefore, $\sum_{i=0}^{K-1} (\epsilon_i + \eta_i) = K$, which implies that $\beta = 1$.

C. Myopic Policy

Consider the special case where $W_{k+1}(\cdot) \equiv 0$ for each k . In other words, for each k , $g_k \in \arg\max_{g \in \mathcal{A}} r_k(x_k, g)$. We call this the *myopic policy*. For the myopic policy, we have that $k = 0, \dots, K-1$,

$$\epsilon_k = 1 - \frac{r_{k+1}(x_{k+1}, g_{k+1})}{r_1(x_1, g_1)},$$

where $x_{i+1} = h_i(x_i, g_i)$ for $1 \leq i \leq k-1$, and

$$\eta_k = \frac{r_{k+1}(x_{k+1}, o_{k+1})}{r_1(x_1, o_{k+1})},$$

where $x_{i+1} = h_i(x_i, o_i)$ for $1 \leq i \leq k-1$. It is clear that because $r_k(\cdot, \cdot) > 0$, we have $\epsilon_k < 1$, in which case $\beta > 0$. In fact, it is easy to check that f is forward monotone with respect to G_k in this case.

D. Rollout of Myopic Base Policy

Consider the rollout method where the base policy is the myopic policy defined above. It is well known that the resulting rollout scheme performs at least as well as the myopic policy [4]. Here, we will calculate a bound on the amount by which the rollout scheme outperforms the myopic base policy in terms of ϵ_k and η_k . This calculation involves introducing some additional notation (which seems unavoidable).

Let $G_K^M = (g_1^M, \dots, g_K^M)$ be the myopic strategy and $G_K^{RM} = (g_1^{RM}, \dots, g_K^{RM})$ the corresponding rollout strategy. More specifically, given g_1^M, \dots, g_{k-1}^M ,

$$g_k^M \in \operatorname{argmax}_{g^M \in \mathcal{A}} r_k(x_k^M, g^M)$$

where $x_1^M = x_1$ is given and $x_{i+1}^M = h_i(x_i^M, g_i^M)$ for $1 \leq i \leq K-1$. Moreover, the rollout scheme with the myopic base policy is as follows: given $g_1^{RM}, \dots, g_{k-1}^{RM}$,

$$g_k^{RM} \in \operatorname{argmax}_{g^{RM} \in \mathcal{A}} \{r_k(x_k^{RM}, g^{RM}) + \sum_{i=k+1}^K r_i(x_i^{RM}, \pi_b(x_i^{RM}))\}$$

where

$$\pi_b(x_i^{RM}) \in \operatorname{argmax}_{g^{RM} \in \mathcal{A}} r_i(x_i^{RM}, g^{RM}),$$

$x_1^{RM} = x_1$ is given, $x_{i+1}^{RM} = h_i(x_i^{RM}, g_i^{RM})$ for $1 \leq i \leq k-1$, and $x_{i+1}^{RM} = h_i(x_i^{RM}, \pi_b(x_i^{RM}))$ for $k \leq i \leq K-1$.

Let f^M and f^{RM} respectively denote the objective functions corresponding to the myopic and rollout (with myopic base policy) strategies. Then we have that

$$f^M((g_1^M, \dots, g_k^M)) = \sum_{i=1}^k r_i(x_i^M, g_i^M)$$

where $x_{i+1}^M = h_i(x_i^M, g_i^M)$ for $1 \leq i \leq k-1$, and $x_1^M = x_1$ is given. Moreover,

$$\begin{aligned} f^{RM}((g_1^{RM}, \dots, g_k^{RM})) \\ = \sum_{i=1}^k r_i(x_i^{RM}, g_i^{RM}) + \sum_{i=k+1}^K r_i(x_i^{RM}, \pi_b(x_i^{RM})) \end{aligned}$$

where $x_{i+1}^{RM} = h_i(x_i^{RM}, g_i^{RM})$ for $1 \leq i \leq k$, $x_{i+1}^{RM} = h_i(x_i^{RM}, \pi_b(x_i^{RM}))$ for $k+1 \leq i \leq K-1$, and $x_1^{RM} = x_1$ is given.

We claim that $f^{RM}(G_1^{RM}) \geq f^M(G_K^M)$. To see this, for the myopic policy, we have

$$g_k^M \in \operatorname{argmax}_{g^M \in \mathcal{A}} r_k(x_k^M, g^M)$$

for $k = 1, \dots, K$. For rollout with the myopic base policy, we have

$$\begin{aligned} g_1^{RM} \in \operatorname{argmax}_{g^{RM} \in \mathcal{A}} \{ & r_1(x_1^{RM}, g^{RM}) \\ & + r_2(x_2^{RM}, \pi_b(x_2^{RM})) \\ & + \dots + r_K(x_K^{RM}, \pi_b(x_K^{RM})) \}. \end{aligned}$$

Because $\pi_b(x_i^{RM}) \in \operatorname{argmax}_{g^{RM} \in \mathcal{A}} r_i(x_i^{RM}, g^{RM})$ and $x_1^M = x_1^{RM}$, we have that

$$\begin{aligned} & r_1(x_1^{RM}, g_1^{RM}) + r_2(x_2^{RM}, \pi_b(x_2^{RM})) \\ & + \dots + r_K(x_K^{RM}, \pi_b(x_K^{RM})) \\ & \geq r_1(x_1^M, g_1^M) + r_2(x_2^M, g_2^M) \\ & + \dots + r_K(x_K^M, g_K^M), \end{aligned}$$

which means that $f^{RM}(G_1^{RM}) \geq f^M(G_K^M)$, as desired.

Combining (11), (12), and the inequality $f^{RM}(G_1^{RM}) \geq f^M(G_K^M)$, we have

$$\begin{aligned} f^{RM}(G_K^{RM}) - f^M(G_K^M) & \geq \left(\sum_{i=1}^{K-1} (1 - \epsilon_i) \right) f^{RM}(G_1^{RM}) \\ & \geq \frac{\sum_{i=1}^{K-1} (1 - \epsilon_i)}{\sum_{i=0}^{K-1} (\eta_i)} f^{RM}(O_K), \end{aligned}$$

which provides a bound on the amount by which the rollout scheme outperforms the myopic base policy.

VI. CONCLUSION

We have developed a systematic approach to deriving guaranteed bounds for approximate dynamic programming (ADP) schemes in optimal control problems. The approach is to formulate a string-submodular optimization problem for which the optimal strategy is the optimal control solution, and the greedy strategy is the ADP solution. Using this approach, we have shown that any ADP solution achieves a performance that is at least a factor of β of the performance of the optimal control solution (satisfying Bellman's optimality principle). The factor β depends on the specific ADP scheme. We have explicitly characterized this dependence and we have illustrated the applicability of our bounding technique to a few examples of ADP schemes, including the popular rollout method.

REFERENCES

- [1] A. A. Ageev and M. I. Sviridenko, "Pipage rounding: A new method of constructing algorithms with proven performance guarantee," *Journal of Combinatorial Optimization*, vol. 8, no. 3, pp. 307–328, 2004.
- [2] S. Alaei and A. Malekian, "Maximizing sequence-submodular functions and its application to online advertising," *arXiv preprint arXiv:1009.4153*, 2010.
- [3] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific, 2000.
- [4] D. P. Bertsekas, J. N. Tsitsiklis, and C. Wu, "Rollout Algorithms for Combinatorial Optimization," *J. of Heuristics*, vol. 3, pp. 245–262, 1997.
- [5] M. Conforti and G. Cornuejols, "Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the rado-edmonds theorem," *Discrete applied mathematics*, vol. 7, no. 3, pp. 251–274, 1984.
- [6] U. Feige and J. Vondrak, "Approximation algorithms for allocation problems: Improving the factor of $1 - 1/e$," in *Proc. 47th IEEE Symposium on Foundations of Computer Science*, pp. 667–676.
- [7] U. Feige and J. Vondrak, "The submodular welfare problem with demand queries," *Theory of Computing*, vol. 6, pp. 247–290, 2010.
- [8] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *Journal of Artificial Intelligence Research*, vol. 42, no. 1, pp. 427–486, Sep. 2011.
- [9] A. Kulik, H. Shachnai, and T. Tamir, "Maximizing submodular set functions subject to multiple linear constraints," in *Proc. 20th ACM-SIAM Symposium on Discrete Algorithms*, 2009, pp. 545–554.
- [10] J. Lee, M. Sviridenko, and J. Vondrak, "Submodular maximization over multiple matroids via generalized exchange properties," *Mathematics of Operations Research*, vol. 35, no. 4, pp. 795–806, 2010.
- [11] E. Liu, E. K. P. Chong, and L. S. Scharf, "Greedy adaptive measurements with signal and measurement noise," in *Proceedings of the Asilomar Conference on Signals, Systems, and Computers*, Asilomar Hotel and Conference Grounds, Pacific Grove, California, November 4–7, 2012, paper TP3a-3, pp. 1229–1232.
- [12] G. L. Nemhauser and L. A. Wolsey, "Best algorithms for approximating the maximum of a submodular set function," *Mathematics of Operations Research*, vol. 3, no. 3, pp. 177–188, 1978.

- [13] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.
- [14] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Hoboken, NJ: J. Wiley & Sons, 2007.
- [15] M. Shamaiah, S. Banerjee, and H. Vikalo, "Greedy sensor selection: Leveraging submodularity," in *Proc. 49th IEEE Conference on Decision and Control*, 2010, pp. 2572–2577.
- [16] M. Streeter and D. Golovin, "An online algorithm for maximizing submodular functions," in *Proc. 22nd Annual Conference on Neural Information Processing Systems*, Dec. 2008.
- [17] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.
- [18] M. Sviridenko, "A note on maximizing a submodular set function subject to a knapsack constraint," *Operations Research Letters*, vol. 32, no. 1, pp. 41–43, Jan. 2004.
- [19] J. Vondrák, "Optimal approximation for the submodular welfare problem in the value oracle model," in *Proc. 40th ACM Symposium on Theory of Computing*, 2008, pp. 67–74.
- [20] J. Vondrák, "Submodularity and curvature: the optimal algorithm," *RIMS Kokyuroku Bessatsu B*, vol. 23, pp. 253–266, 2010.
- [21] Z. Wang, W. Moran, X. Wang, and Q. Pan, "Approximation for maximizing monotone non-decreasing set functions with a greedy method," *preprint*.
- [22] Z. Zhang, E. K. P. Chong, A. Pezeshki, and W. Moran, "String submodular functions with curvature constraints," in *IEEE Trans. Automatic Control*, submitted Jun 2013.
- [23] Z. Zhang, Z. Wang, E. K. P. Chong, A. Pezeshki, and W. Moran, "Near optimality of greedy strategies for string submodular functions with forward and backward curvature constraints," in *Proceedings of the 52nd IEEE Conference on Decision and Control*, Florence, Italy, December 10–13, 2013, pp. 5156–5161.